DATABASE DESIGN

Other Dependencies and SQL

University of Waterloo

List of Slides

Other Dependencies

other anomalies/redundancies in relational schemas?

⇒ beyond FDs and BCNF??

Example: consider the following table (in BCNF):

Course	**T**eacher	**B**ook
Math	Smith	Algebra
Math	Smith	Calculus
Math	Jones	Algebra
Math	Jones	Calculus
Physics	Black	Mechanics
Physics	Black	Optics

⇒ tuples (Course, Set-of-profs, Set-of-books) in 1NF.

Multivalued Dependencies (MVD)

- CTB table contains redundant information because:

  whenever $(c, t_1, b_1) \in CTB$ and $(c, t_2, b_2) \in CTB$
  then also $(c, t_1, b_2) \in CTB$
  and, by symmetry, $(c, t_2, b_1) \in CTB$

- we say that a **multivalued dependency** (MVD)

  $C \twoheadrightarrow B$ (and $C \twoheadrightarrow T$ as well)

  holds on CTB.

---

given a course, the set of teachers and the set of books are uniquely determined and independent.

Another Example

Course	**T**eacher	**H**our	**R**oom	**S**tudent	**G**rade
CS101	Jones	M-9	2222	Smith	A
CS101	Jones	W-9	3333	Smith	A
CS101	Jones	F-9	2222	Smith	A
CS101	Jones	M-9	2222	Black	B
CS101	Jones	W-9	3333	Black	B
CS101	Jones	F-9	2222	Black	B

- FDs:
  $C \to T$, $CS \to G$, $HR \to C$, $HT \to R$, and $HS \to R$
- MVDs:
  $C \twoheadrightarrow HR$

Axioms for MVDs

1. $Y \subset X \Rightarrow X \twoheadrightarrow Y$ (reflexivity)
2. $X \twoheadrightarrow Y \Rightarrow X \twoheadrightarrow (R - Y)$ (complementation)
3. $X \twoheadrightarrow Y \Rightarrow XZ \twoheadrightarrow YZ$ (augmentation)
4. $X \twoheadrightarrow Y, Y \twoheadrightarrow Z \Rightarrow X \twoheadrightarrow (Z - Y)$ (transitivity)
5. $X \rightarrow Y \Rightarrow X \twoheadrightarrow Y$ (conversion)
6. $X \twoheadrightarrow Y, XY \rightarrow Z \Rightarrow X \rightarrow (Z - Y)$ (interaction)

Theorem: Axioms for FDs (1)-(6) are sound and complete for logical implication of FDs and MVDs.

Example

In the $CTHRSG$ schema $C \twoheadrightarrow SG$:

1. $C \twoheadrightarrow HR$
2. $C \twoheadrightarrow T$ (from $C \rightarrow T$)
3. $C \twoheadrightarrow CTSG$ (complementation of (1))
4. $C \twoheadrightarrow CT$ (augmentation of (2) by C)
5. $CT \twoheadrightarrow CTSG$ (augmentation of (3) by T)
6. $C \twoheadrightarrow SG$ (transitivity on (4) and (5))

$\Rightarrow$ the “rest” of the $CTHRSG$ attributes (except C)
are **partitioned** to disjoint groups
each *multidetermined* by C.

Dependency Basis

Definition:

A **dependency basis** for X with respect to a set of FDs and MVDs F is a partition of $R - X$ to sets $Y_1, \ldots, Y_k$ such that $F \models X \twoheadrightarrow Z$ if and only if $Z - X$ is a union of some of the Y_is.

- used as a quick test of $F \models X \twoheadrightarrow Y$ and $F \models X \to Y$
- unlike for FDs we can't split right-hand sides of MVDs to single attributes (cf. minimal cover).
- the dependency basis of X w.r.t. F can be computed in PTIME [Beeri80].

Lossless-Join Decomposition

- similarly to the FD case we want to decompose the schema to avoid anomalies

  $\Rightarrow$ a lossless-join decomposition (R_1, R_2) of R
  with respect to **MVDs** F:

$$F \models (R_1 \cap R_2) \twoheadrightarrow (R_1 - R_2)$$

  or, by symmetry

$$F \models (R_1 \cap R_2) \twoheadrightarrow (R_2 - R_1)$$

- this condition implies the condition for FDs (in only FDs appear in F).

Fourth Normal Form (4NF)

Definition:

Let R be a relation schema and F a set of FDs and MVDs.

Schema R is in **4NF** if and only if

whenever $(X \twoheadrightarrow Y) \in F^+$ and $XY \subseteq R$, then either

- $(X \twoheadrightarrow Y)$ is trivial ($Y \subseteq X$ or $XY = R$), or
- X is a superkey of R

A database schema $\{R_1, \ldots, R_n\}$ is in BCNF if each relation schema R_i is in 4NF.

$\Rightarrow$ use BCNF-like decomposition procedure to obtain 4NF.

Example

The CTB schema can be decomposed to 4NF (using $C \twoheadrightarrow P$) as follows:

Course	Teacher
Math	Smith
Math	Jones
Physics	Black

Course	Book
Math	Algebra
Math	Calculus
Physics	Mechanics
Physics	Optics

$\Rightarrow$ no FDs here!

Other Dependencies

- **Join Dependency** on R

  $\Rightarrow \bowtie [R_1, \ldots, R_k]$ holds if $I = \pi_{R_1}(I) \bowtie \ldots \bowtie \pi_{R_k}(I)$

  $\Rightarrow$ generalization of an MVD

  $X \twoheadrightarrow Y$ is the same as $\bowtie [XY, X(R-Y)]$

  $\Rightarrow$ **cannot** be simulated by MVDs

  $\Rightarrow$ no axiomatization

  $\Rightarrow$ Project-Join NF (5NF)

  $\bowtie [R_1, \ldots, R_k]$ implies R_i is a key.

- **Inclusion Dependency** on R and S

  $\Rightarrow R[X] \subseteq S[Y]$ holds if $\pi_X(I_R) \subseteq \pi_Y(I_S)$

  $\Rightarrow$ relates **two** relations

  foreign-key relationships

Integrity Constraints vs. Queries

Observation:

- Queries:

  first-order formulas true in a database (instance) for a given valuation.

- Integrity constraints:

  assertions about all valid database instances

Integrity Constraints as queries?

$\Rightarrow$ **YES**, we use *closed first-order formulas*

Dependencies as Formulas

Let $R(A, B, C)$ and $S(A, B)$ be two relational schemas.

- FD $A \to B$ on R:

  $\forall x, y_1, y_2, z_1, z_2 . R(x, y_1, z_1) \wedge R(x, y_2, z_2) \Rightarrow y_1 = y_2$

- MVD $A \twoheadrightarrow B$ on R:

  $\forall x, y_1, y_2, z_1, z_2 . R(x, y_1, z_1) \wedge R(x, y_2, z_2) \Rightarrow R(x, y_1, z_2)$

- IND $R[A] \subseteq S[A]$ on R and S:

  $\forall x, y, z . R(x, y, z) \Rightarrow \exists u . S(x, u)$

- etc, etc.

Generalized Dependencies

Why don’t we just use general first-order formulas for integrity constraints?

- notational convenience
- decidable theories

  . . . at least in many cases.
- axiomatizations
- normal forms induced by the constraints

Integrity Constraints in SQL

- connected with a table (definition)
  - ⇒ Primary Keys
  - ⇒ Foreign Keys
  - ⇒ **CHECK** constraints
- separate ECA rules (triggers)

Primary Key

- specifies a *primary key* in a table
- syntax:

```
CREATE TABLE <name>
   (   ... <attributes>,
     PRIMARY KEY ( <list of attr> )
   )
```

- also creates an unique index on the key

Example

```
create table DEPT
 ( ID            integer not NULL,
   DeptName      char(20),
   MgrNO         char(3),
   PRIMARY KEY (ID)
 )
```

Example (cont.)

```
sql => insert into DEPT values \
sql (cont.) => ( 1 ,'Computer Science', 000100)
DB20000I  The SQL command completed successfully.

sql => insert into DEPT values \
sql (cont.) => ( 1 ,'Computer Science', 000100)
SQL0803N  One or more values in the INSERT or UPDATE
statement are not valid because they would produce
duplicate rows for a table with a unique index.
SQLSTATE=23505
```

Foreign Key

- specifies an *referential constraint*
- syntax:

```
CREATE TABLE <name>
 (   ... <attributes>,
   FOREIGN KEY ( <attrs> )
    REFERENCES <ref-table>( <attrs> )
    ON DELETE <delete-action>
    ON UPDATE <update action>
```

- the actions can be:
  - ∗ **RESTRICT** – produce an error
  - ∗ **CASCADE** – propagate the delete
  - ∗ **SET NULL** – set to “unknown”

Example

```
create table EMP
 ( SSN     integer not NULL,
   Name    char(20),
   Dept    integer,
   Salary  dec(8,2),
   primary key (SSN),
   foreign key (Dept) references DEPT(ID)
                      on delete cascade
                      on update restrict)
```

Example (cont.)

```
db2 => insert into EMP \
sql (cont.) => values ( 999, 'DAVE', 2, 50000 )
SQL0530N  The insert or update value of FOREIGN KEY
"DAVID.EMP.SQL970916001756640" is not equal to any
value of the primary key of the parent table.
SQLSTATE=23503

db2 => insert into EMP \
sql (cont.) => values ( 999, 'DAVE', 1, 50000 )
DB20000I  The SQL command completed successfully.
db2 => select * from emp where SSN=999

SSN         NAME                 DEPT SALARY
----------- -------------------- ---- ----------
        999 DAVE                    1   50000.00

db2 => delete from DEPT where id=1
DB20000I  The SQL command completed successfully.
db2 => select * from emp where SSN=999

SSN         NAME                 DEPT SALARY
----------- -------------------- ---- ----------
```

CHECK constraints

- allow checking for “correct” data:
- syntax:

```
CREATE TABLE <name>
   (   ... <attributes>,
       CHECK <condition>
   )
```

- condition is a *simple* search condition
  ⇒ no subqueries (in DB2)

Example

```
create table EMP
 ( SSN     integer not NULL,
   Name    char(20),
   Dept    integer,
   Salary  dec(8,2),
   primary key (SSN),
   foreign key (Dept) references DEPT(ID)
                 on delete cascade
                 on update restrict,
   check ( salary > 0 )
 )
```

```
db2 => insert into emp values (998, 'DAVE', 1, 0 )
SQL0545N  The requested operation is not allowed
because a row does not satisfy the check constraint
"DAVID.EMP.SQL970916000939620".  SQLSTATE=23513
```

Active (ECA) Rules

- for more complex integrity constraints and other things
- general structure:

```
CREATE TRIGGER <name>
  <event> ON <table>
  REFERENCING <transition-tables>
  FOR EACH ROW | FOR EACH STATEMENT
  WHEN <condition> <sql-statement>
```

- events:
  - **AFTER** or **NO CASCADE BEFORE**, and
  - **INSERT**, **DELETE**, or **UPDATE** (col)
- transition tables:
  - **OLD AS <id>**, **NEW AS <id>** (single row)
  - **OLD_TABLE AS <id>**, **NEW_TABLE AS <id>** (transition tables)

Summary

Schema design summary:

1. Create an ER diagram

   $\Rightarrow$ visualization of the design goals

2. Translate ER-to-Relational

3. Determine FD, MVD, JD, . . .

   $\Rightarrow$ detect anomalies and decompose

   $\Rightarrow$ find **keys**

4. Determine inter-relational constraints

   $\Rightarrow$ INDs and foreign key constraints

5. Enforce rest of constraints

   $\Rightarrow$ **CHECK** declarations

   $\Rightarrow$ ECA rules (only as the last resort!)